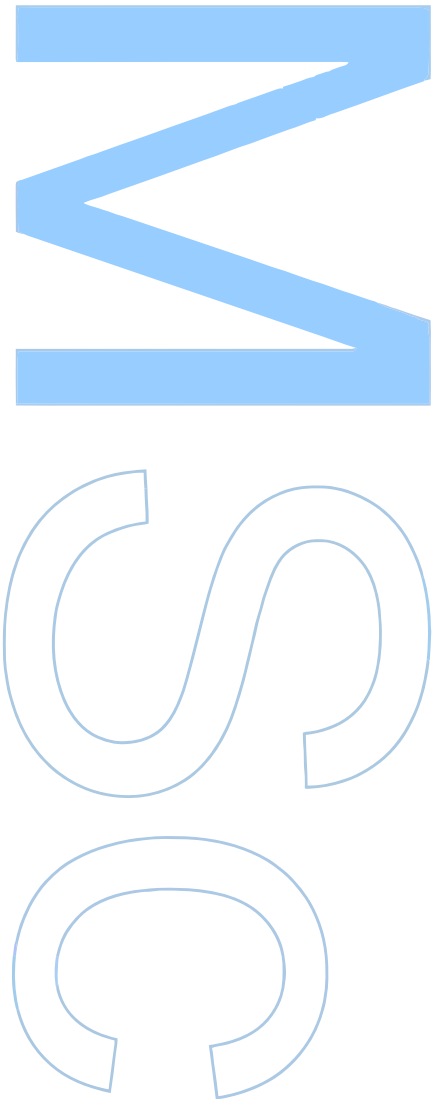
Crowdsourcing Video Replays Using Mobile Edge-clouds

Filipe Esteves
Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos
Departamento de Ciência de Computadores
2017

Orientador
Fernando Silva, Professor Catedrático,
Faculdade de Ciências da Universidade do Porto

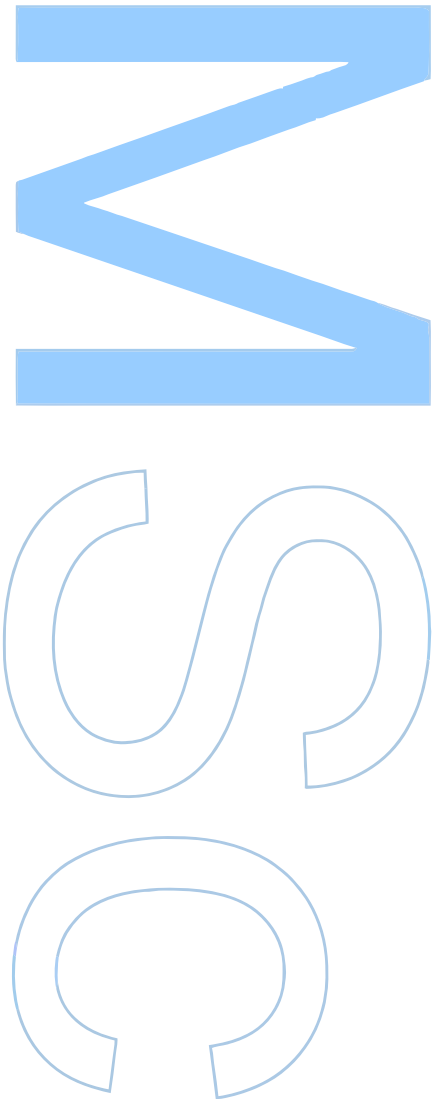
Co-Orientador
Luís Lopes, Professor Associado,
Faculdade de Ciências da Universidade do Porto



Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/ _____/ _____



Acknowledgements

I want to thank my work colleague João Rodrigues who introduced me to the project and helped me in the beginning of my work.
This work was funded by a grant from the Hyrax project (CMUP-ERI/FIA/-0048/2013,FCT).

Resumo

O número de dispositivos móveis aumentou significativamente nos últimos anos e como resultado, a capacidade de armazenamento e processamento disponível dos mesmos aumentou proporcionalmente. Uma grande parte do tempo, estes dispositivos encontram-se em *standby* o que faz com que se tornem num potencial recurso de computação.
Se possuirmos uma grande quantidade de dispositivos móveis próximos uns dos outros com boas capacidade de processamento, podemos conectá-los e criar uma *edge-cloud* local que vai por conseguinte permitir que uma nova gama de aplicações possa ser executada. Estas redes ajudam a resolver duas consideráveis classes de problemas. Primeiro, num cenário de catástrofe, onde as aplicações de *crowdsourcing* podem permitir que dispositivos relativamente próximos possam comunicar sem a necessidade de acesso a um ponto central de controlo. Segundo, num evento muito lotado como por exemplo um jogo de futebol onde hoje em dia a quantidade de pessoas que possuem um dispositivo móvel capaz de partilhar conteúdos multimédia através de uma rede *wireless* é significativa, a infra-estrutura necessária para proporcionar uma boa qualidade de serviço é dispendiosa inclusive em certos casos impossível de implementar. Em relação ao segundo cenário, este género de redes e aplicações de *crowdsourcing* ajudaria a reduzir fortemente a carga colocada nos pontos de acesso de uma infra-estrutura *wireless* (WiFi ou 3G) onde a troca de dados se passaria a dar directamente entre dispositivos (*peer-to-peer*). Assim, não só se libertam recursos do ponto de acesso, deixando este mais livre para executar outras tarefas, como também ajuda a diminuir as latências de comunicação entre dispositivos.
Neste trabalho desenvolvemos uma aplicação Android que amplia a funcionalidade de uma já existente projectada para ser usada no contexto de eventos sociais e que faz uso do *middleware* Hyrax. Permite *caching* e partilha de conteúdos multimédia entre os dispositivos constituintes de uma *edge-cloud*. As características principais adicionadas à aplicação foram a possibilidade de os utilizadores acederem a ambos, eventos em tempo real e deferidos, assim como a possibilidade de diferentes *edge-clouds* poderem trocar conteúdos entre si.

Abstract

The number of mobile devices has largely increased in the past years. As a result the amount of storage and processing power available for people to use increased proportionally. Most of the time these devices are idle and that turns them into a potential resource of computation and storage.
As a consequence of the above people started to notice the power that these devices could have when put together. If we happen to have a significant amount of mobile devices close to each other in a small geographical area with good processing capabilities, we can connect them and create a local edge-cloud and use the corresponding edge-cloud to perform a new range of functional tasks. These networks could help to solve two major problem classes. First, in a catastrophe scenario where crowdsourcing applications may enable relatively close devices to communicate among each other without the need of a central point of control. Second, in a very crowded event, such as a soccer match, where the number of people owning a mobile device capable of sharing multimedia content over a wireless network is large, the infrastructure needed to provide those people with good quality of service would be too expensive and sometimes impracticable to implement.
Regarding the second scenario, this genre of networking and crowdsourcing applications would strongly help to reduce the load placed at the access points of a wireless infrastructure (WiFi or 3G) by sharing the data directly between devices using a peer to peer device communication approach. This way, we not only release access point resources making them available for other possible tasks, but can help to reduce communication latencies among devices.
In this work we developed an Android application which extends the functionality of an existing one designed to be used in the context of social events. It allows for content caching and sharing inside an edge-cloud. The main features added to the application were the possibility for the users to access not only real time events but also deferred events and the possibility of content sharing among different edge-clouds. The new developed application makes use of the Hyrax middleware.

Contents

List of Tables

List of Figures

Acronyms

3G Third generation of wireless mobile telecommunications technology

4G Fourth generation of wireless mobile telecommunications technology

API Application Programming Interface

GO Group Owner

GPS Global Positioning System

iOS iPhone Operation System

IP Internet Protocol

MANET Mobile Ad-hoc Network

MMPI Mobile Message Passing Interface

MPI Message Passing Interface

OSI Open Systems Interconnection

PDA Personal digital assistant

RPC Remote procedure call

TDLS Tunneled Direct Link Setup

UGR User-generated Replays

Chapter 1

Introduction

Over the last years the number of mobile devices increased significantly as well as the amount of usage people gives them [2, 3]. Today's mobile devices have more computational and storage resources than the last decade's desktop and server processors [4]. The new devices are now able to perform a huge set of tasks due to their high processing power, memory and sensing capacities. Nowadays smartphones are able to run demanding applications in terms of processing power and memory, such as games, Internet browsing, applications capable of displaying high definition audio and video as VLC Player [5], as well as applications that make use of the sensing capacities like health monitoring and geolocation applications as Runtastic [6], Glucose Buddy [7] and iCare Blood Pressure Monitor [8].

In this context we can view mobile devices as "thick clients" or "thin servers" instead of "thin clients" as they have always been seen [9].

With the proliferation and enhanced capabilities of mobile devices, it's now acceptable to see a wireless cloud of nearby smartphones as an interesting collective computational/storage resource. There is an immense range of new applications that can result if users are able to pool their smartphone's data and processing power with those of others in their proximity as the Alljoyn framework [10], Apple's Implementation of Bonjour [11] and FireChat [12].

It is also possible for users to get alternative and richer sources of information from the surrounding environment, making use of nearby device functionalities which allows them to get a more accurate, real-time picture of the local unfolding events.

A mobile edge-cloud consists of a group of mobile devices interconnected between them using a networking protocol, e.g., (ad-hoc network) networks in order to pool their storage and computational resources for creating such a network that abstracts

all the ongoing communications and seems like a unified, single and powerful device to the applications. Edge-clouds can be helpful in two major problem classes: when there is no connection at all to a major infrastructure (usually the Internet) and when the current infrastructure is not enough to support all the traffic generated by the hosting of a large event. Figure 1.1 shows an example scenario of an earthquake where the use of edge-clouds for communication and information sharing could help to facilitate people's situation and in some cases save lives. When a large event such as a soccer match takes place the number of users who attend the event is usually massive which in turn makes the conventional wireless structures unable to deal with all the traffic generated in these situations. If we want to provide the users with an infrastructure capable of dealing with such a massive flow of data and quality of service at the same time the costs to implement such structure would be unbearable in most cases.
There are some issues associated with the edge-cloud architecture that we didn't have to deal with in the standard cloud architecture. Among them churn, is a characteristic of mobile networks and is known as the node join and leave of the network. When we are talking about a network formed by mobile devices which have limited radio interfaces and power supply that communicate among each other via radio signals this problem becomes an important issue to tackle. There are some security concerns and privacy issues that arise in the edge-cloud architecture, issues that do not exist in the standard cloud infrastructure but in this work we are not going to address them. In this work we are going to implement and demonstrate an Android application (User Generated Replays) that uses the Hyrax middleware. The Hyrax project [13], provides a middleware that give programmers an interface that allow for crowdsourcing applications to be built upon edge-clouds. The User Generated Replays is one of the case studies of the Hyrax project and aims to provide users the possibility to record a video using his own smartphone and make it available for the nearby devices, at the same time, allow the user to see videos that others recorded and made available in the same event. This is achieved by sharing the user's generated data with peer-to-peer mechanisms and local caching as will be explained in detail later on.

1.1 Motivation

A large event is usually characterized by a large affluence of people. A soccer match can hold tens of thousands of fans watching the game, so the number of mobile devices by that time is proportionally large. Due to the fact that nowadays soccer stadiums are in some cases giant infrastructures, though they are designed specifically to hold

Figure 1.1: Hearthquake scenario

tens of thousands of people there are some issues that need to be tackled in order to provide all those people attending the event with the same experience. People who sit close to the field can have a better view of the game than those who sit far from the field and those whose view is partially blocked by some structure or object. In order to allow those people to see in more detail what they might have not seen clearly or might completely lost, the idea of video recording and sharing comes handy. A fan can record a move or other development, and if he finds it interesting enough to share, he can make it available to all the event subscribers, so a person who was not able to see that specific move or development can watch it on his smartphone and then continue to follow the event. This idea rises an issue that could make it impracticable. The content sharing infrastructure needed to support such applications in events like a soccer match. In order to enable mobile devices to communicate and share content among each other securely and fast in a scenario like the previously described we need a wireless infrastructure. The cost of a wireless infrastructure capable of supporting those kind of content sharing applications in such events is sometimes impracticable or enormous as we can see from the American Superbowl [14]

case, where they implemented a colossal wireless infrastructure to deal with the traffic generated. In events like the Superbowl the concept of edge-cloud really comes in handy. An edge-cloud would allow users to share content in a peer-to-peer fashion way and by using local caching the load placed in the access points of the infrastructure would be diminished. This would provide users with a much better quality of service once it could diminish significantly content sharing latencies and release at the same time the access point resources for other possible uses. If we make use of edge-clouds in such events the size of the infrastructure needed to support all the traffic generated might not be something with implausible costs to implement.
In this work we focused in extending the already existing application by rewriting it now upon the Hyrax middleware and adding new functionalities as: the possibility for different edge-clouds connected to the same Access Point to share content between them, the possibility to engage now multiple events at once as deferred events and real time events all at the same time, a Web Interface to allow the administrators to control the settings and contents of the available events. Figure 1.2 shows as an example of how the edge-cloud infrastructure could be disposed in a soccer stadium.

Figure 1.2: A public event where edge-clouds can be valuable for content sharing

1.2 Problem Statement

Given the User Generated Replays scenario we ask which would be the main hindrances regarding the application implementation upon the Hyrax middleware.
The Hyrax platform provides us a middleware that abstracts not only all the physical hardware control under the network layer as well as it provides an interface with some network layer algorithms to access and control the work we can perform inside the cloud.
We need to create an application that not only works as a full proof concept of the Hyrax edge-cloud architecture, but also makes, at the same time, use of the full potential of the edge-cloud architecture features.
The UGR is able to record video and make it available so that any user connected to that same cloud is able to see the available videos and download it entirely to his device if and only if he wants to do so based on the replay's description and thumbnail. The user should be able to decide which videos he will make available for sharing as well as to choose which videos from the whole list of available videos he wants to download and keep in his device. The content dissemination is supposed to be done by wireless communication methods as WiFi, WiFi Direct, TDLS. The data is supposed to be disseminated among devices directly in peer-to-peer way.
The application must not only be able to create, manage and control all the edge-cloud network formed, but also to manage all the multimedia contents, stored in the device and available for download. The interface must be easy to use and clear as possible for the user.
The main overall goals for the application are, firstly, the user should be able to download the multimedia contents at least as fast as he would if he was using a standard wireless network infrastructure and second it should remove a generous amount of load from the infrastructure access points to which the edge-cloud is connected.
The possibility for different edge-clouds in the same network to share contents between them is not yet explored and it's a subject that we'll explore in detail and test in this work.
The user must be able to access various events, including real time events as well as already deferred events using the Android UGR application. We also propose to create an administrator web interface to allow the administrators to control the events' settings, the available contents in each event, create new events and insert or delete replays from an already existing event.

1.3 Work Plan

In order to adequately proceed to implement the application, we firstly need to study the state of the art of the application, particularly the Hyrax middleware upon which our application will be implemented. Second, we must study the already existing specification of the application and look for ways to improve and expand it. The UGR application will be a mobile application and will be built for the Android operating system. It will be implemented upon the Hyrax platform which is a middleware that not only abstracts all the distinct communication interfaces of Android devices, but also implements the network layer functionalities necessary to support the edge-cloud. Hyrax structural organization is divided in layers.
The User Generated Replays App will interact with the network layer of Hyrax which is where all the network logic and algorithms remain. The application will communicate with Hyrax' network layer through a well-defined programming interface.
Lastly, we need to test our application in an adequate and suitable event for the circumstance. We must verify if our overall goals for latency and access points' load reduction is verified as well as if the application is working properly without any major issues.

1.4 Thesis Layout

In Chapter 1 we introduced our work by describing the idea behind our project, what could be done and improved and the way how we plan to do that. Chapter 2 describes the state of the art, what has already been done so far and the existing applications and technologies that share some conceptual principles with ours. In Chapter 3 and 4 we describe the already existing User Generated Replays application and the functionalities and improvements we made over the existing one. In Chapter 3 we focus more on explaining and detailing the already existing application and in Chapter 4 we focus on our contribution to that application. In Chapter 5 we show the experiments and measurements we did as well as we analyze the obtained results.

Chapter 2

State Of The Art

In the last few years with the development of new technologies and the evolution of mobile device's storage and computational power new computing architectures have been proposed. The concept of Edge-Cloud/Edge-Computing, Mobile Cloud Computing and Fog Computing they differ in some key aspects. In order to get a more concrete idea of the current status of these concepts we must define them and explore the work that has been done, what is being done and find the possible fields that we can explore and do some research on.

2.1 Mobile Cloud Computing

There are several definitions for Mobile Cloud Computing which may vary depending on the subject being explored. The most commonly accepted and used definition for Mobile Cloud Computing means to run tasks of an application such as Google's Gmail for Mobile on a remote server. Due to the fact that the mobile devices are less computationally capable and have several limitations, sometimes its a good idea to perform computationally expensive tasks on a server with plenty of resources to do that. In this case the mobile device acts like a thin client that connects itself to the remote server and then offloads data and computation (the task content) to the server in order to get the task result as an outcome. There are plenty of services that make use of this scheme and some more examples are Twitter for Mobile and Facebook. This becomes handy mainly because mobile devices have a very limited supply of energy which drains quickly when performing heavy computation and they

are slow when compared to personal computers and servers.
Using this paradigm of computation the only work of a mobile device is to offload the tasks to the server and then gather the outcomes and process them. The burden of the computation is performed in the server. The device must connect to the server through Wi-Fi or mobile network such as 3G or 4G when available. Sometimes it is not the most appropriate approach to use since the power drained with the communication is greater than the power the device would consume if it performed the task locally. In cases where the app requires low latency or when the amount of data that is needed to transfer between the device and the server is too large, and the computation that the server will perform is small the offload of the task from the mobile device to the server is not the best approach to use.

2.2 Edge Clouds

Although in previous work, scientists tried to offload computation from the mobile device to the cloud in order to save energy and release the limited device resources for other parallel computations, Rodriguez et al. [15] suggest that the power of interconnected mobile devices can lead to good results when performing heavy and computationally expensive tasks. Even though current mobile applications treat the end-user as a "thin-client" as previously stated and due to the ongoing development and evolution of mobile devices, it begins to make sense to gather the interconnected power of these devices and to avail that power to perform expensive computations.

MMPI

Since its appearance, the *Message Passing Interface* [16] proved to be a success for inter-node communication in clusters and parallel machines. The conversion of that concept to the mobile world was proposed and tested by Doolan et al. [17]. They presented the MMPI *Mobile Message Passing Interface* for mobile devices. MMPI was implemented in Java and the authors used Bluetooth for communication between devices. In this work, the authors used Bluetooth to form a peer-to-peer network in order to spread messages, but they do not provide methods to deal with churn.
The discovery process proved to be the most time consuming element of the network formation procedure, whether in laptops or PDA's. The discovery times increased

as more devices were found as the time needed for service discovery and network formation.

FemtoCloud

FemtoCloud [18] proposes a concept of cloud computing that shares a lot of principles with the edge-cloud. It uses a network of mobile devices to perform computation. There is what the author calls the *Control Device* which controls and distributes jobs to other nodes in the network (clients). Every client is running a client service and is supposed to estimate the computational capacity of the device and use it along with the input in order to estimate the available resources for the possible execution of future assigned jobs. The client leverages essentially device sensors information, user input and utilization history so that the *Control Device* can create a node profile and assign tasks accordingly. The authors propose a client software that interacts with the control device, by providing relevant information for job allocation, as well as receiving and generating work. The authors tested the prototype application developed against an Oracle cloud under two different scenarios, a full presence scenario, and an emulated arrival/departure scenario. In this experiment, the authors use three devices, a Galaxy S5 running Android 4.4.4 in addition to a Nexus 10[2013] and a Nexus 7[2013] tablets running Android 5.0.2. They compare the performance of FemtoCloud to an oracle which assumes accurate knowledge of all connectivity and execution time for every task on every device. In the first scenario, the authors assume that the three devices existed during the whole period of experiment (1 hour). In the second scenario they emulate average presence time of two minutes for each device. FemtoCloud achieved approximately 85% of what the Oracle cloud achieved in both scenarios.

mCloud

mCloud [19] presents an idea for of a cloud composed of mobile devices inter-connected mostly by WiFi networks and other short range radio communication methods in order to share storage and computational resources and execute assigned tasks. The main motivation for the creation of this platform was to decrease the amount of mobile data used. To perform the tasks mCloud can choose to do it locally, only if it can be performed by a single device, or distributed, in which case the task is divided by the master node and shared among the group of devices willing to perform those tasks (slaves). The master node is the one who performs the network formation, divides

the task and assigns the tasks to be performed to the *slave* nodes who are willing to perform them.

Honeybee

Honeybee [20] is a framework, that uses croudsourcing to perform computations that enable humans to perform qualitative classification tasks such as conducting surveys, asking for opinions, image identification and comparison which involves the collaboration of multiple human users using mobile devices. In this framework Android devices form a Mobile Ad-hoc network (MANET) using either Bluetooth or Wi-Fi Direct on which the Master/Group Owner is responsible for scheduling and assigning the tasks accordingly to each Slave/Group Member. In combination to the scheduler, nodes in this network acquire jobs using Work Stealing. When using Bluetooth an initial set of jobs is sent to the nodes that will be computing during the formation. On Wi-Fi Direct, this does not happen, jobs are distributed during execution time. This framework also provides some failure tolerance techniques.
The authors implemented three applications using the Honeybee framework to evaluate the performance and feasibility. Those were a distributed face detection, distributed mandelbrot set generation, and collaborative photography. All the three applications tested demonstrated considerable speedups especially when the job size was larger. The authors also showed that a small group of powerful devices achieves a better performance speedup than a large group of relatively weaker devices.

Bonjour

Bonjour [11] makes it easy to discover, publish, and resolve network services by using a sophisticated and easy-to-use programming interface that is accessible from Cocoa, Ruby, Python, and some other languages. The services provided by Bonjour are divided into three main areas: Addressing (allocating IPs to Devices), Naming (creating aliases for each network device) and Service Discovery. This implementation is open source, and contains implementations for OS X, iOS, Linux and Windows. The discovery works in a publish-subscribe way. The nodes advertise services and those become visible to the other nodes on the network so they can use them. Bonjour is also known as zero-configuration networking, enables automatic discovery of devices and services on a local network using industry standard IP protocols.

Firechat

FireChat [12] is a proprietary mobile app, developed by Open Garden, that allows smartphones to organize into a wireless mesh network and exchange messages using technologies such as Bluetooth or Wi-Fi. It allows to form a mesh network, making possible the peer-to-peer connectivity without having an explicit Internet connection. The Firechat app offers a secure multi-user and multi-room chat with flexible features, such as authentication, moderator capabilities, user presence and search, private messaging and chat invitations. Those services are provided using distinct wireless technologies, e.g. Bluetooth, infrastructure Wi-Fi, peer-to-peer Wi-Fi and Apple's multipeer connectivity Framework [21] (only available for Apple devices).

Alljoyn Framework

Alljoyn [10] is an open source network framework that allows different types of devices and apps to discover and communicate among each other in an abstract way, this means hiding the inherent complexity of the distinct underlying network protocols and hardware. Generically, it publishes APIs over the network through a general bus, which permits distinctive network technologies to be used, such as Wi-Fi, Wi-Fi Direct, Bluetooth, Ethernet and PowerLine. Regarding the network formation, Alljoyn, uses a super peer paradigm, mesh of stars network, where the leaf nodes are connected to router nodes and these act as bridges to the others router nodes.

2.3 Crowd Sourcing

Crowdsourcing is a distributed problem solving model in which a call to an undefined number of people is made in order to engage that same people in the process of solving a complex problem. Even though crowdsourcing is not fully adapted yet to the mobile phone workforce the widespread use of smartphones will reveal the full potential of this new problem solving approach. Smartphones offer a great platform for extending existing Web-based crowdsourcing applications to a larger contributing crowd, making contribution easier and omnipresent. Furthermore, smartphones' multisensing capabilities as movement, geolocation, light, audio and visual sensors offer a variety of efficient new ways to collect data, enabling new crowdsourcing applications.

Context-Aware mobile Crowd Sourcing

The involvement of citizens in public decision making is now becoming a trend in areas such as urban planning and quality assessment campaigns of public services. In [22], the authors describe mechanisms that they designed to incentive smartphone users to participate in mobile phone sensing which is a new paradigm that allows us to collect and analyze sensed data far beyond what was previously possible. They considered two different models of sensing mechanisms. They named those two models, one the "platform-centric model" and the other "user-centric model". On the platform-centric model the reward is shared by the users involved while on the user-centric model, users can ask for a reserve price for its sensing service.
For the platform-centric model, they modeled the incentive mechanism as a Stackelberg game in which the platform is the leader and the users are the followers. This enables the platform to maximize its utility while no user can improve its utility by deviating from the current strategy. For the user-centric model, they designed an auction mechanism, which they called MSensing. They proved that the mechanism is computationally efficient, meaning that the winners and the payments can be computed in polynomial time (linear) which makes it scalable. The mechanism is also what they called "individually rational" which means that each user will have a non-negative utility, profitable because the platform will have no loss and lastly truthful, meaning that no user can improve its utility by asking for a price different from its true cost.

Participatory Sensing Crowd Sourcing

In [23] the authors introduce a novel paradigm called participatory sensing for achieving large-scale urban sensing by crowdsourcing sensing data from the mobile phones of ordinary citizens.
Participatory sensing offers a number of advantages over traditional sensor networks, which entails deploying a large number of static wireless sensor devices, particularly in urban areas. First, since participatory sensing leverages existing sensing (mobile phones) and communication (cellular or WiFi) infrastructure, the deployment costs are virtually zero. Second, the inherent mobility of the phone carriers provides unprecedented spatiotemporal coverage and also makes it possible to observe unpredictable events (which may be excluded by static deployments). Third, using mobile phones as sensors it's a cheap option for the majority of users to acquire sensors.

2.4 Hyrax

The Hyrax project proposes a new concept where a cloud constructed only by mobile wireless devices with the purpose of pooling its computational and storage resources together in order to support a new set of proximity-aware applications that benefit the owners of those devices. All the constituent nodes of the cloud are edge (non server-caliber) computers, and all the computation is performed completely within the edge-cloud, i.e, there is no offloading of data and computation to third parties like the traditional cloud infrastructure. Because sometimes a typical smartphone does not have the necessary resources and processing power to perform a certain task alone, or at least to perform it in a decent amount of time, with a significant amount of users a cloud of mobile devices can more easily find a solution to that particular problem.
Hyrax is a middleware structured in layers. Hyrax is still in development and the objective is to provide not only an API for communication between devices, but also the construction of a logic network and other services like RPC, storage and video streaming. Hyrax abstracts all the complexity of the network operations behind the API, so that the developer only needs to set up a few operations and the middleware will handle all the logic behind the process. The figure 2.1 shows the structure of the middleware.
The bottom layer of Hyrax, the corresponding Data Link Layer on the OSI model will abstract all the communication interfaces of the different devices and provide a clean interface for the upper layers to control it. For the developer using Wi-Fi, Wi-Fi TDLS, Wi-Fi Direct or Bluetooth is unimportant, the methods provided in the interface to control them are the same. The layer implementation handles all the logic behind each technology. The second layer is the network layer which is responsible for handling the formation and management of the logic network built on top of the physical network where all the devices are connected. Then the overlay layer which comes on top of the networking layer allows for the creation of groups on top of the network formed by the network layer. Lastly the services layer is where the implementation of common network services come such as streaming and storage.

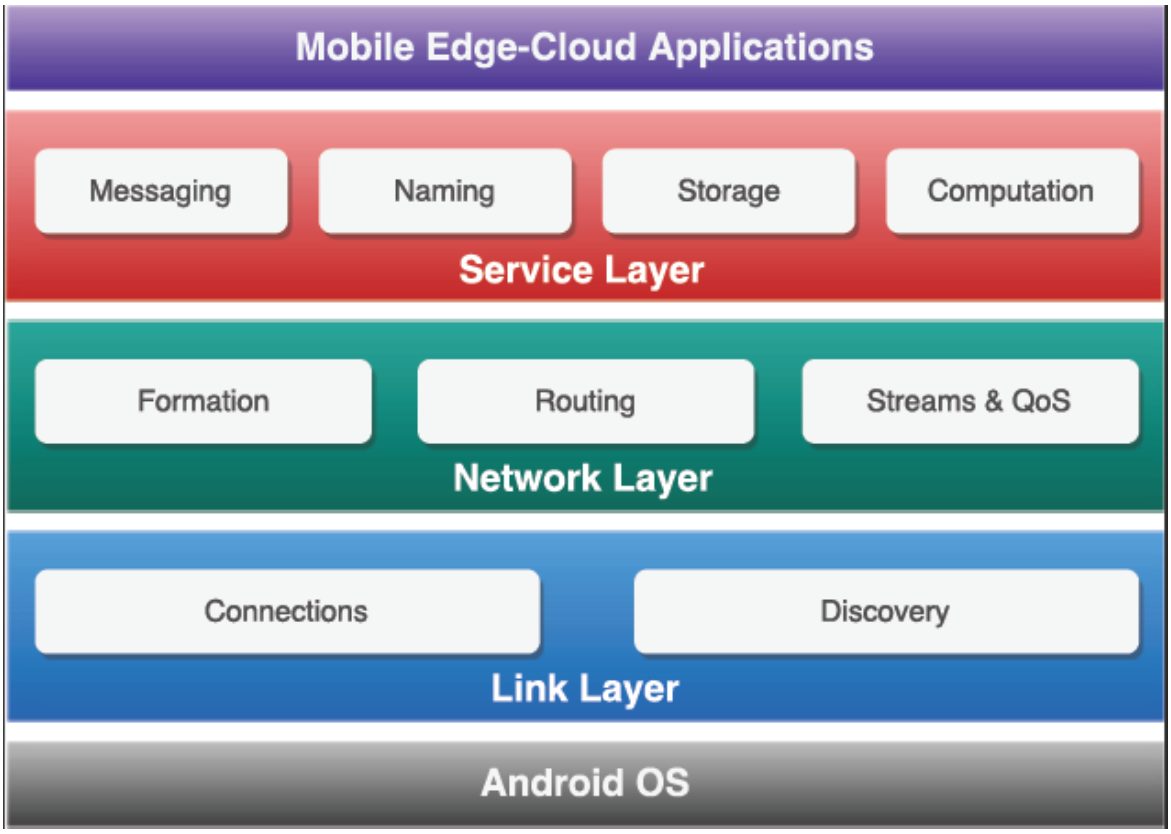


Figure 2.1: The Hyrax middleware structure design (adapted from [1])

Chapter 3

UGR Primary architecture

In a social event as a soccer match the possibility for people to see again the game's dubious tosses in their own smartphone without the need to use internet access was the main purpose of the primary version of UGR. The first version made no use of the Hyrax middleware and it was completely built from scratch with a very small set of functions. The users were supposed to install an Android application on their mobile devices in order to make use of the edge-cloud functionalities and use the provided distribution content platform. This architecture was composed of three major components: a Remote Server, a Group Owner and the Peer as you can see in Figure 3.1.

3.1 The Remote Server

In this architecture the server's main role was to provide to all the connected devices videos/replays and associated metadata of a certain event. The server was the remote device that contained the database and worked as service provider to all the devices that requested the replays and their associated metadata. A server could only serve one event at a time and all the videos were stored on it.

3.2 The Group Owner

The Group Owner was the central mobile device on the client side. Its role was to serve not only as a bridge between the server and all the other devices but it worked

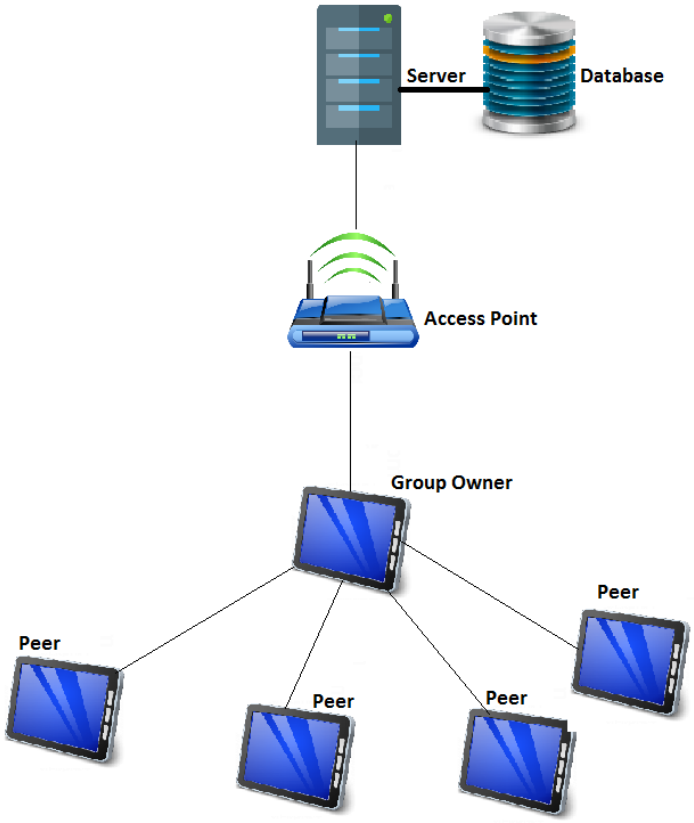


Figure 3.1: Primary Architecture

as well as a regular client at the same time. The Group Owner is the device that its simultaneously connected to the Remote Server through the access point and to all the other devices. The Group Owner acts as a client to the Remote Server and as a server to all the mobile devices that are connected to it (peers). It is able to be connected to all those devices and access point simultaneously because we were using devices that support dual-band Wi-Fi communication. In wireless networking, dual-band equipment is capable of transmitting in either of two different standard frequency ranges. Modern Wi-Fi home networks feature dual-band broadband devices that support both 2.4 GHz and 5 GHz channels in two different interfaces. The Group Owner was the device responsible to ensure the communication between the Remote Server and all the other devices. Whenever a device needed to download a video from the Remote Server the connection was made through the Group Owner. It contained not only all the metadata about the network (edge-cloud) but also, all the information about the replays each device stored in cache.

3.3 The Peers

Peers are all the devices that make up the edge-cloud of which the Group Owner is also part. As said before the Group Owner is a device that acts simultaneously as a peer and as a server. All the peers run the same Android Application as well the same distributed network algorithm. The peers are the ones who contain cached replays. The peers all together may not contain a full copy of all the replays existing in the Remote Server and every peer may not have the same exact replays stored in its memory as the others. Even the whole edge-cloud may not have all the replays cached. Peers are an end terminal for the whole system. The peers are the user's devices so to speak. The peer is supposed not only to give the end user the possibility to search for available replays on the network and download them but also to reproduce them. The peer is connected to the Group Owner and might be directly connected to another peer on the same group if they are sharing a file between them by using Wi-Fi TDLS. In this case the peer is connected to both the Group Owner and the other peer.

3.4 Edge-cloud Formation

The Remote Server is the only place where all the event's replays are stored before the edge-cloud formation starts to take place. All the mobile devices run the same exact application. The first device to connect to the access point and subsequently to the Remote Server will take the role of Group Owner. After it assumes the role of Group Owner it starts to advertise to the network its presence so the next device that connects to the same network and starts the UGR application will firstly search for existing Group Owners and only then it will connect to one of them if there are any available. It will continuously behave the same way on and on every time a new device enters the network and wants to join the edge-cloud. Once the Group Owner reaches a point where it cannot accommodate more connections, the new device becomes a new Group Owner which in turn will start a new connection to the Remote Server and form a new edge-cloud. Every new device that wants to join the edge-cloud will now connect to the device that has room for new connections and answers the request for a new connection first.

3.5 Content Flow Dynamics

Once the Group Owner is connected to the server it downloads all the event's metadata. The event's metadata include event's start and end time (if available), event name, location and event's description. It will download as well all the replays metadata which includes information about the list of replays as its name, duration and description. The Android application will then display to the user the list of available replays for download. The user could then chose which ones to download and do it directly from the server if he wants to. Once a new device enters the edge-cloud the first thing it will try to do is look for available Group Owners and connect to one of them. After the connection is established it will download from the Group Owner the event's metadata and replays metadata as well. In case the Group Owner does not have this information already the Group Owner will download it from the Remote Server and only then it will provide all the requested data to the newly connected device. The new device will now have access to the list of available replays and all its associated information. If the user chooses to download a certain replay it will first ask the Group Owner for the existence of that same replay on the network. In case the video is already cached in the edge-cloud the video is not downloaded from the Remote Server, instead it will be downloaded directly from the device containing it on the edge-cloud. The connection is established through the Group Owner. The Group Owner will serve as a bridge between the two devices. The replay's contents will go thought the Group Owner from the device containing it to the device requesting it. The Group Owner does not keep of any of the forwarded contents. The amount of devices that can be simultaneously connected to the same Group Owner depends not only on the kind and model of devices used but also on the interference existing in the environment in that same exact moment.

Chapter 4

UGR Extended Architecture

4.1 Main Role Components

The original UGR architecture was revised and a large space for expansion and implementation of new functionalities arose. A new Android application was built from scratch, this time upon the Hyrax middleware. The three main role players kept being the same (Remote Server, Group Owner, Peers), but a new set of functionalities were added to the original idea. Those functionalities are:

- Enable Peers to upload videos
- Possibility to access multiple events, deferred and real-time
- Content sharing among different edge-clouds connected to the same access point
- UGR Web administrator interface to modify event's settings and contents

4.2 Events

Once we want to make use of the UGR platform in a social event like a soccer match, a conference or even a party a new event should be created. By default all the available events in the Remote Server are subscribed by the peers automatically at once. After subscribing all the events, people can then chose to open an event and start recording and uploading new videos to the Remote Server in order to make it available for all the other people attending the event. Every replay is associated only to a certain

event, for instance, once you open a certain event you can only see videos uploaded for that same event. Events can only be created and managed by administrators using the Web Application interface. An event can be closed or open. A closed event is an event where users can no longer add more videos, it's locked as opposed to an open event where users can still upload and insert new replays on the network. Each event has a name, a description, a location, a start and end time.

4.3 Replays

A replay is a video who was either previously recorded by someone who was using the UGR edge-cloud and made available to all the people subscribing the event or a video introduced by the administrator directly in the Remote Server (using the web application). They are both equal in terms of structure but they come from different sources. If the administrator wants to, he can supervise and block all the videos users uploaded that are inappropriate for sharing, otherwise they are exactly the same. Each replay has a name, a description, a video and a thumbnail associated.

4.4 The Remote Server

The Remote Server was completely built using Node.js and it was designed to serve two ends. To serve the edge-cloud's requests (Android App) and Web Application requests. All the replay's videos, thumbnails and associated metadata are stored on the Remote Server as well as the open events information. All the metadata is stored in a MongoDB database and the videos are stored in a separated folder this one inside the server too. Regarding the replay's thumbnails and videos the database only stores its local address, all the information contained in the database is stored as text, there's no Blobs (Binary large objects) inside of it. Although the server is configured to serve the same content to the Android application and to the Web Application the way it delivers the content is slightly different. The contents to the Web Application are delivered by HTTP while to the Android application it we use a Google technology called protocol buffers [24]. Protocol buffers are a language-neutral, platform-neutral extensible mechanism for serializing structured data which come in handy for dealing with this kind of data dissemination among different platforms.

4.4.1 The Database

The database stores all the events, replays and users' metadata. The database has three collections with the following structure:

```
db.users
{
        "_id" : ObjectId("5953c86eb3b64c002b942e87"),
        "email" : "master@dcc.fc.up.pt",
        "password" : "fancyPassword"
}
```

The users collection stores system administrators credentials for authentication.

```
db.replays
{
        "_id" : ObjectId("5953c86fb3b64c002b942e88"),
        "name" : "cf9750f751b495bf_f4d610ad",
        "eventShortId" : "cf9750f751b495bf",
        "thumbName" : "cf9750f751b495bf_f4d610ad.jpeg",
        "thumbLength" : 3364,
        "length" : 10962469,
        "extension" : ".mp4",
        "fullname" : "cf9750f751b495bf_f4d610ad.mp4",
        "duration" : 4,
        "legend" : "Replay description",
        "n_downloads" : "0",
        "n_views" : "0",
        "supervised" : "true"
}
```

The replays collection contains a list of objects, one for each replay existing in a given event. Each stores the event ID, the video file ID which corresponds to its own name on the filesystem besides the file extension, the thumbnail file name which is the same as the video file name except the file extension part, video size in bytes, video file extension, video duration in minutes, a replay description, some statistics regarding the number of times it was downloaded and viewed and finally a field indicating if the replay was supervised or not. The administrator can chose to supervise all the replays filtering the ones who are inappropriate to disseminate. Since the database does not store the files, those are stored in regular system directories which includes the video files and thumbnails. The videos from all events are all in the same folder put together, the same happens to the thumbnails, which are in a separate folder from the videos, but the thumbnails of the events are all put together too.

```
db.events
{
        "_id" : ObjectId("5953c86eb3b64c002b942e83"),
        "name" : "President Marcelo",
        "description" : "President Marcelo visits FCUP on FCUP day",
        "shortId" : "cf9750f751b495bf",
        "dateInit" : ISODate("2016-10-07T12:00:00Z"),
        "dateEnd" : ISODate("2016-10-07T17:00:00Z"),
        "location" : "FCUP",
        "closed" : false
}
```

The events collection is a set of objects each one for every existing available event. It contains the event name, the event description, the event ID, start and end time, location and finally a tag specifying if the event is closed or not.

4.5 Group Owner

The Group Owner as in the primary architecture is the device that connects the edge-cloud to the Remote Server. It was now redesigned to allow communication among groups. The edge-cloud is made by the Group Owner plus the set of devices who are connected to it (Peers). It is now possible to share content among different edge-clouds if their Group Owners are connected to an access point and they are all in the same network. The Group Owner has now a table not only containing a list of all the cached replays inside the edge-cloud but also the contents existing in other edge-clouds connected to the same network. Regarding this is now possible to see a set of edge-clouds connected to the same network as a single larger edge-cloud. The Group Owner has information regarding its own edge-cloud network formation. It has now a proxy, it enables peers to establish a connection to the Remote Server and Group Owners from other different edge-clouds in the same network.

4.6 Peer

Peers have now almost the same set of capabilities they had in the previous architecture except they are now able to record videos and upload them to the Remote Server and search for cached replays outside of its own edge-cloud. The main purpose of the peers is still the same, its to give the user an interface to interact with the edge-cloud,

download replays and watch them. The Group Owner acts as a peer at the same time. Usually the Group Owner is the first device to get connected to the Remote Server.

4.7 Network Formation

Once the Remote Server is active and functional devices can then start looking for it to connect and start downloading available videos. First, devices connect to the access point and then they start looking the network for available Group Owners to connect to. The Group Owners periodically broadcast the network with information announcing its presence and availability to the other devices. The new device will listen the network for available Group Owners and if after a certain period of time the discovering device doesn't receive any broadcast announcement from the Group Owner it becomes a Group Owner. Once a device becomes a Group Owner it starts advertising the network for it's presence and availability so the upcoming devices can become aware of the existence of an already active and available Group Owner. Once a Group Owner reaches the limit for the number of available peers it could hold it stops advertising its presence to the network. The next upcoming device will have to take the role of Group Owner again. The peers only communicate directly with the Group Owner except when two peers are connected to the same GO, in this case they can establish a WiFi TDLS connection between them. When two devices under the same network need to exchange content and that transfer is made using another device as intermediate, WiFi TDLS can have a huge role regarding the content transfer. WiFi TDLS allows two devices to connect directly to each other and transfer content directly among themselves without making so much use of the intermediary node. This only happens when the two devices are close enough to establish a direct WiFi connection between them. We can have multiple edge-clouds connected to the same access point inside the same sub-network. If it happens that the Group Owners of two different edge-clouds are under the same sub-network (connected to the same access point) the edge-clouds can then share content between them without making any use of Remote Server which removes some load from it. This is a functionality that was added in this newest version which will be explained in more detail later.

4.8 Content Dissemination

The Remote Server is the main storage device. Once the first Group Owner connects to the Remote Server the first thing it will request is the list of available events. The server answers by sending to the newly connected device the list of available events it can subscribe. The user chooses among the available events one to engage in. The next thing the user's device will do is to ask the server for the list of replays existing in that specific event. The server will then answer with a package containing the list of available replays in that event plus all the metadata associated with each replay like description and title. The client will then request the server for the replay's respective thumbnails one by one. Once it downloads all of them from the server it will present the user the list of available replays which the user can choose to download or not. The Group Owner periodically requests the Remote Server for an updated version of the replays list.

4.8.1 When Second Device Engages in the Network

After a new device joins the edge-cloud it will take the role of peer and connect to the existing Group Owner. It will ask the Group Owner for the available events list. The Group Owner will usually answer the peer with a package containing a list of all the open events available. If this process fails the peer will request the Remote Server for the events list. This request will not be a direct request, instead the request will go through the Group Owner once it is connected to it. The Remote Server will then answer the peer with a package containing a list of available events. Once the device owner selects an event it will ask the Group Owner for the list of available replays in that event plus all the respective thumbnails. If the Group Owner has this information it will send all the requested information and data to the peer. If the Group Owner doesn't have such content the peer will ask the Remote Server for it, following the same process it did previously with the event list. In this case the Remote Server will answer the peer with the list of available replays in that specific event plus the replay thumbnails.

4.8.2 Device to Device Replay Transfer

Once a user requests to download a replay the first thing his device will have to do is question from where it will download the contents. The replay can be either

downloaded from within the group, another group or the server. To do that the device will send to the Group Owner a request asking for a list of devices from where is it possible to download the replay. The Group Owner will then answer with a message containing a list of the devices containing the replay either inside or outside the group. If the replay the user wants to download is already cached inside the same edge-cloud it will try to get it first from the group. If not, the device will try to download it from another group and lastly if all the previous options fail the device will resort to the server. The Group Owner is the only device in the edge-cloud who knows which devices have the replays cached and which ones either inside or outside the group. If a device in the same group has cached the replay the user wants, the device will try to download the replay from it first. It will make three attempts to download it. If all of them fail, it will perform three attempts again to download the replay from outside the group (only if anyone has the replay cached in another group). The GO is the only device in the edge-cloud who knows where the cached replays are stored including the ones in other edge-clouds that are connected to the same Access Point. Lastly, if after three attempts the user cannot get it from another group it will recur to the server and download it from there directly.

4.8.3 The Client Server architecture

Every device runs a client and a server. All devices run the same server and the same client, the difference is that not every device is able to answer certain requests or at least give an updated answer besides the Group Owner. For instance, only the Group Owner can answer requests regarding which device has a certain cached content because is the only device containing that information or it is the only one who can provide an updated list of replays to the Peers, even if the peers can do it too, it is not necessarily an updated version. The Group Owner has a proxy running which serves as bridge for the peers to establish a direct connection to the Remote Server or to other devices outside of the network. It only allows the establishment of connections from the inside to the outside and not the other way around. When a peer requests a Remote Server for a certain replay, the request goes through the Group Owner but the process is completely transparent for the group owner. The proxy running in the Group Owner will forward the peer's request to the Remote Server without any modifications to the original content as well the answer from the server to the peer. It is thanks to the proxy that runs in the Group Owner that a peer from one group can directly request the Group Owner from another group for a certain replay. The peer only need to have its IP address and it will be completely transparent.

4.9 The Android Application

Once we connect to the access point and open the UGR application this is the first screen it shows (if it connected successfully to the Remote Server) showing all the available events the user may engage in. The user can now chose among the available events which one he wants to engage in, download and upload replays to. The network formation and content dissemination is completely transparent to the user. After a user chooses which event he wants to enter, the list of available replays will appear on the screen. The user can then choose which ones he wants to download based on the replay description and thumbnail and he can then play the downloaded replays on his device. There is a field which allows the user to search for a certain replay based on the its description. The record button that appears on the right side on the top of the list forward the user to another screen where it can record and upload a video or he can chose among the list or recent videos which one to upload as we can see in the Figure 4.2

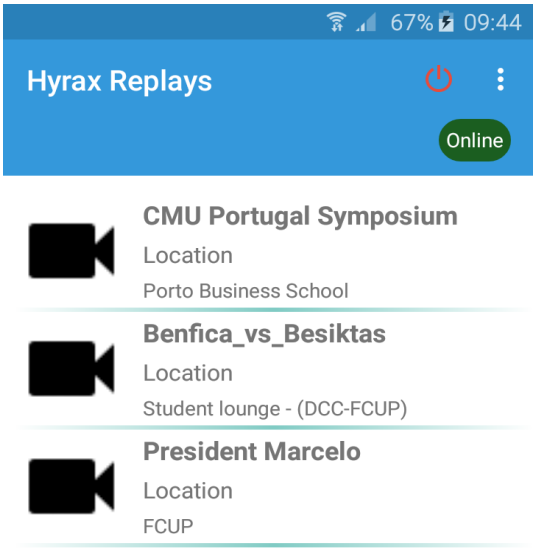


Figure 4.1: Event List screen

Figure 4.2: Replay List screen

The screen we can see in Figure 4.3 allows the user to select the video he wants to upload and make available for the other users as well as to record a new one and share it. It is possible to play the video we want to upload inside the application in order to figure out if it is the one we want to upload or to check if the quality of the video is acceptable. Once we select a video to upload we must insert a video description in order to make it easy for the other users to understand what kind of content it may have and only then we can upload the video.

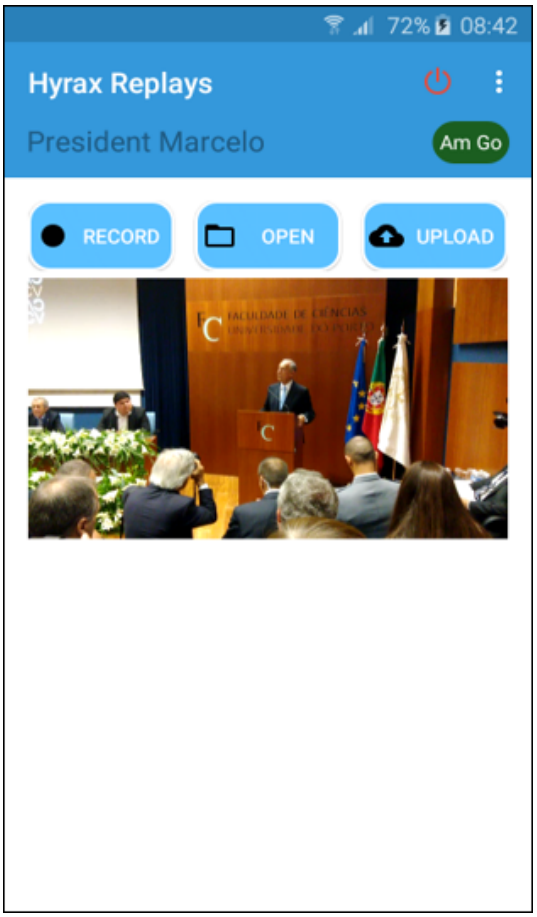


Figure 4.3: Video upload screen

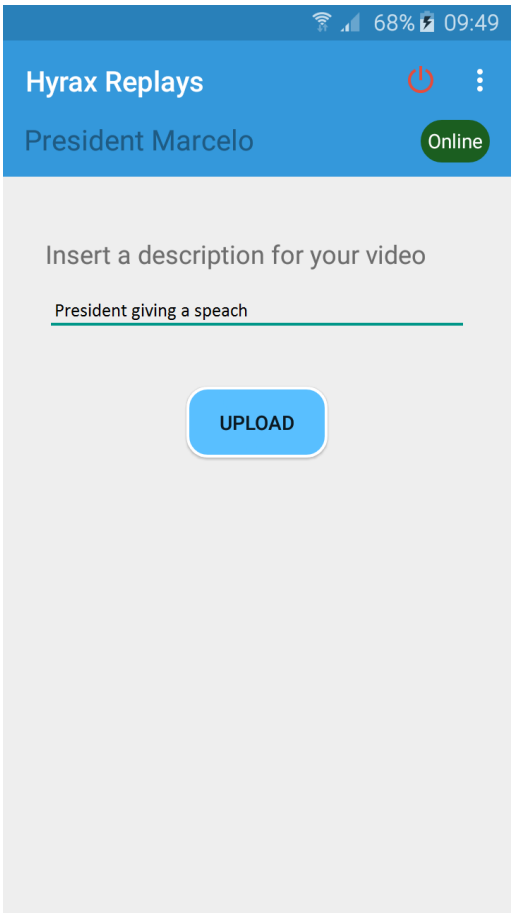


Figure 4.4: Video caption screen

4.10 The Web Interface

The administrator web interface allows the administrators to control the events and its contents. It's possible to: create new events, modify event settings, insert new replays in the database, modify replay settings, and reproduce and supervise replays. Figure 4.5 shows the opening administrator interface. It displays the current available events in the Remote Server/Database. It is in this page that we can chose to modify the event settings of some event and create a new event.

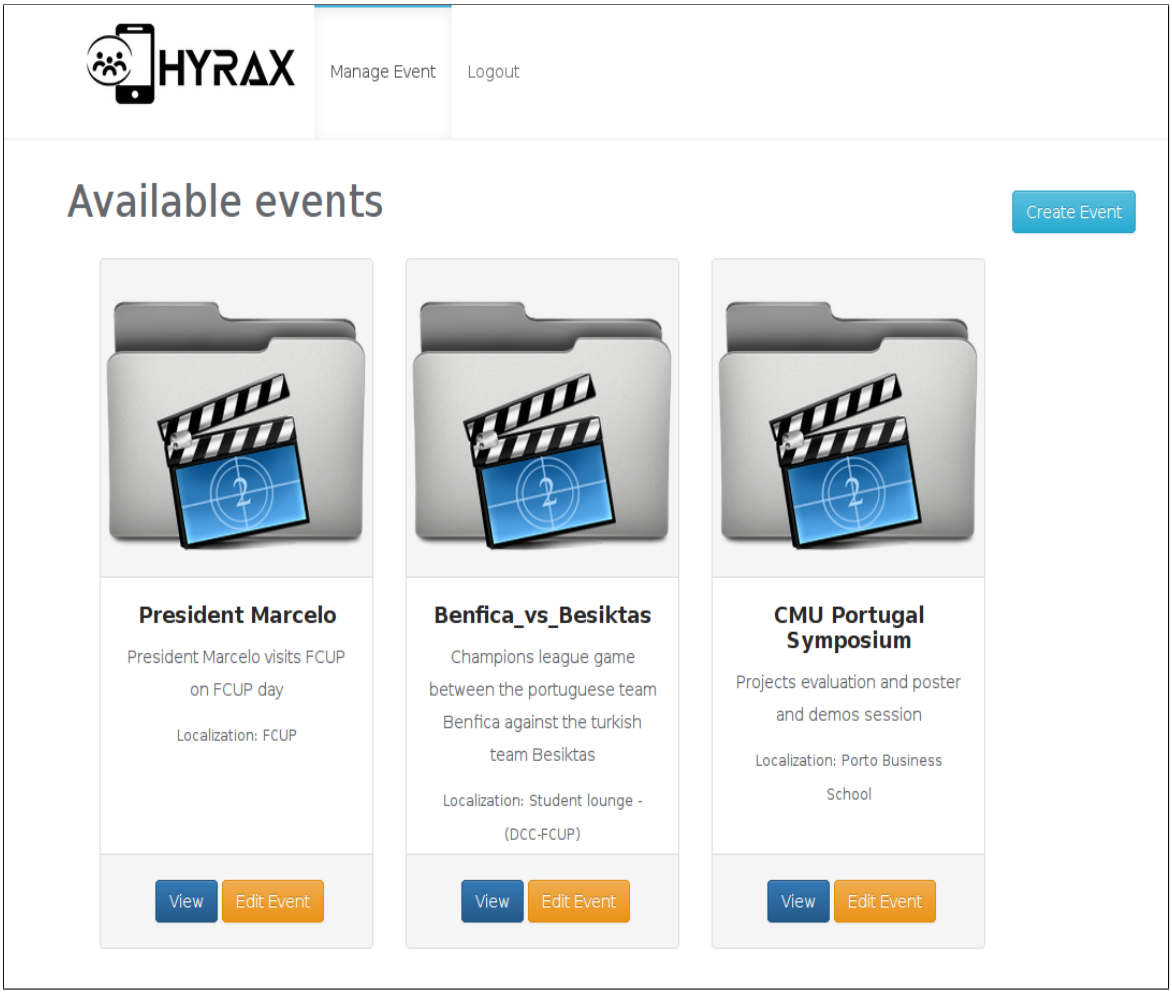


Figure 4.5: Event List Administrator Interface

To create a new event we only need to insert the event's info as: name, location and description in the dialog box.

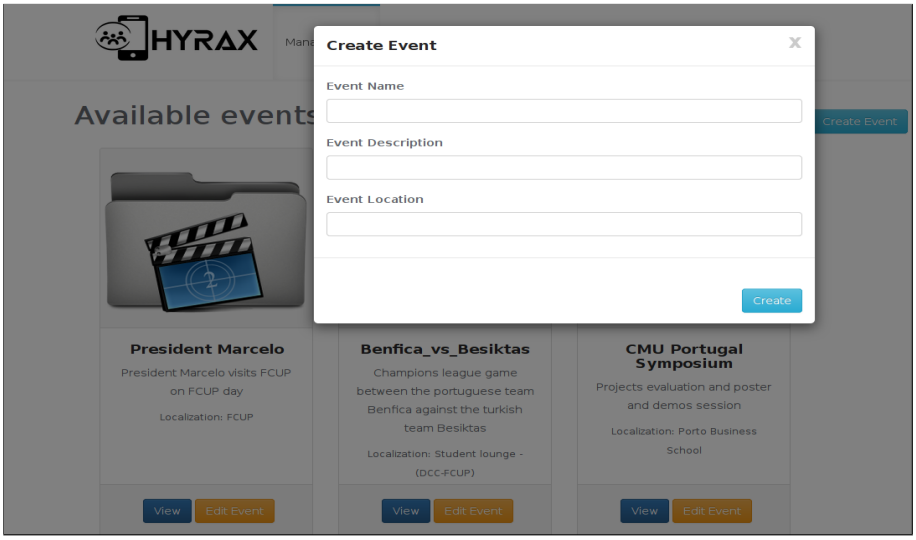


Figure 4.6: Create Event Administrator Interface

In the "Edit Event Configuration" dialog box we can modify the event info as well as open or close the event for the public to upload new videos or even delete the event. We can open this dialog box by clicking on the "Edit Event" button of the respective event.

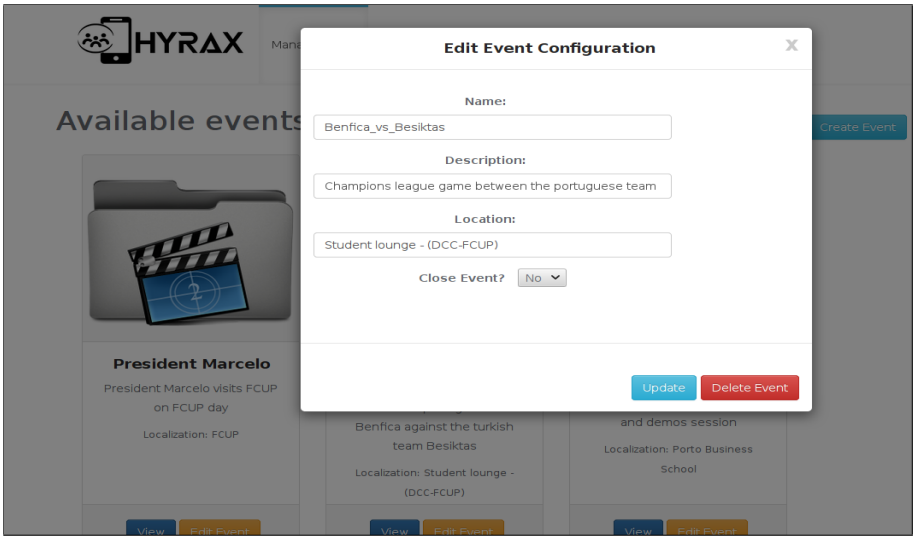


Figure 4.7: Event Settings Administrator Interface

Once we open an even in the administrator interface we can then click on the button "Upload File" and a dialog box with the necessary controls to do it appears on the screen. To insert a replay in the database and make it available for everyone we only need to select the video file we want to upload and insert a replay description to be easier for the user to chose among the list of available replays which to download and reproduce in his device.

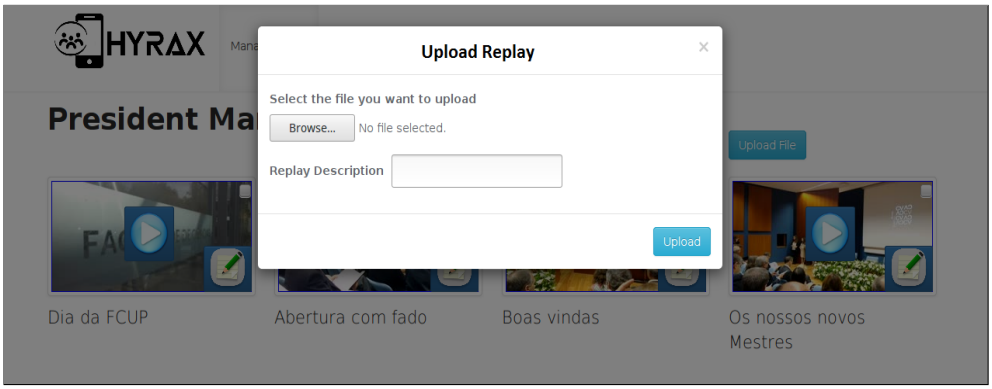


Figure 4.8: Upload Replay Administrator Interface

In the "Edit Replay" dialog box we can mark a replay as supervised or not or even delete it if we do not want it to be available.

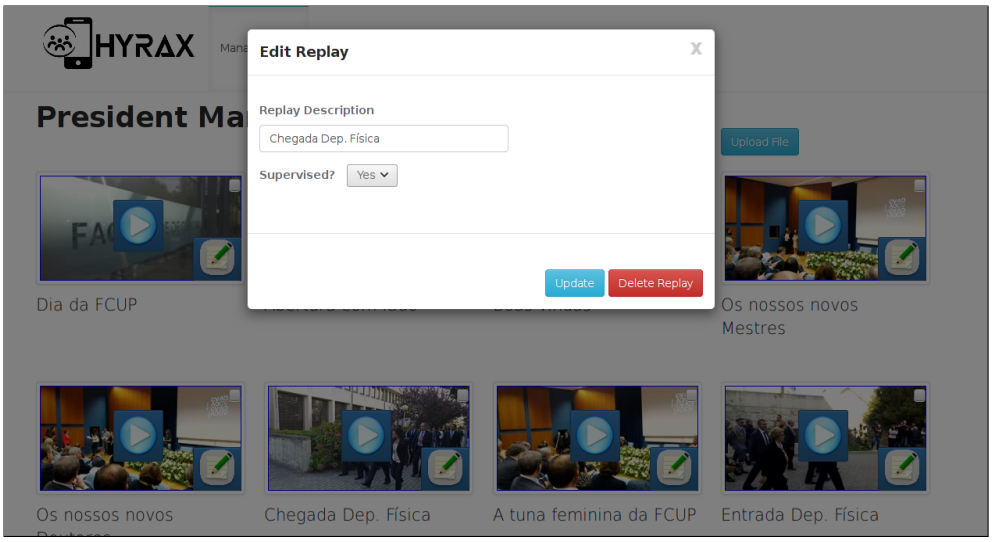


Figure 4.9: Replay Settings Administrator Interface

Last it's also possible to reproduce the replays in the web interface which makes it easier for the administrator to supervise them.

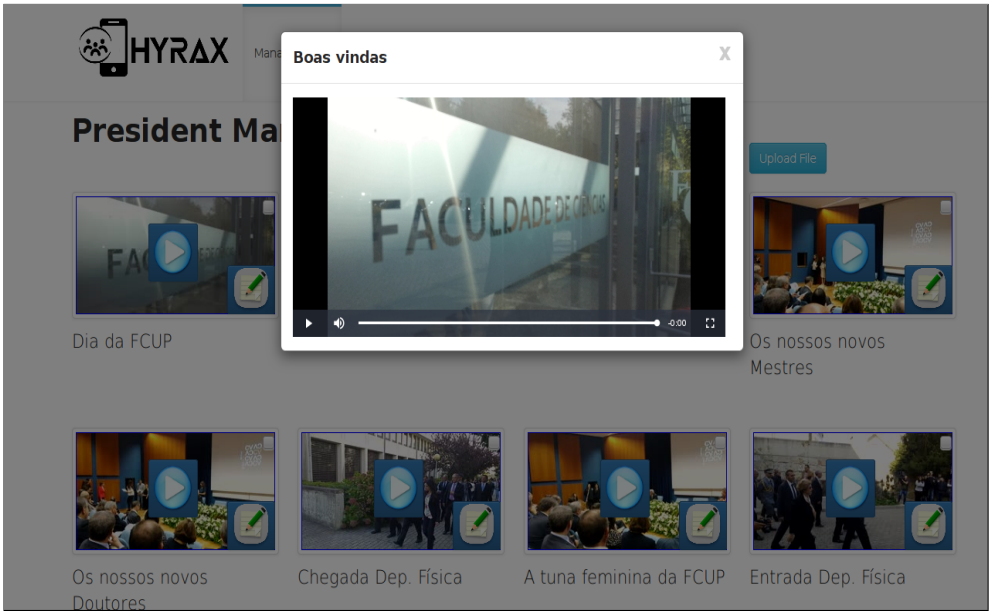


Figure 4.10: Reproduce Replay Administrator Interface

Chapter 5

Inter-Group Communication in UGR

The possibility for two different edge-clouds to communicate was one of the main aims of our work. The communication between different edge-clouds allows us to resort a smaller number of times to the remote server because we can try to fetch the contents we want in other edge-clouds. This way we end up with a larger size cache, from where we can try to fetch the replays and we managed to gain some more independence from the Remote Server that could easily be a central point of failure as well as a bottleneck for the whole system.

5.1 Implementation

The Group Owners who are connected to the same access point all belong to the same network and they can communicate and switch messages among them. This enables us to take use of that functionality in order to allow peers in different groups to communicate and share content. The devices that connect the different groups are their respective Group Owners. All the exchanged traffic among Peers from different groups goes through their respective Group Owner. All the Group Owners have a list of all the replays cached in other groups. Once a peer requests the Group Owner for the list of devices who have a certain replay it answers them with the list of devices who own the replay inside of the group plus the IP address of the Group Owners of other groups who have the replay. It does not necessarily mean that the Group Owner from the other group is the device that has the replay cached, it could be a peer inside

of that group. If nobody in the same group as the device who made requested has the replay and it exists in other groups it will ask the Group Owner of the group who actually has the replay cached for it. The peer requesting the replay will send its request directly to the Group Owner of the group who has the replay cached. This happens because every group owner is running a proxy which allows a peer to contact any device outside the edge-cloud. The Group Owner of the other group will do the rest of the job. It will look up his local tables to know which device has the replay cached and start a direct connection among that device and the Group Owner of the device who requested the file. The peer answering the request will send its answer to the destiny's Group Owner directly. This happens again because of the proxy running in every Group Owner. The destiny's Group Owner will then forward the contents to the peer who requested it. Once the replay is transferred the connection is closed. The motivation for this is to alleviate some load from the Remote Server and at the same time give more independence from it to all the edge-clouds. If the different edge-cloud could share their cached content among they do not need to rely that much on the Remote Server which gives the mobile devices more independence even though it places some extra load on the access points. The Group Owners broadcast periodically the network with information regarding his own edge-cloud and the cached contents it has inside. This is how it is possible for the Group Owners of the other groups to know the contents that exist outside of the group.

5.2 Experiments

After all the infrastructure for content sharing inside the edge-cloud and between different edge-clouds has been set up we measured the transfer rates among devices either in the same group or between edge-clouds. We measured the network transfer rate for 6 different scenarios as we will see next. For each scheme we performed 4 different types of tests. The tests involved downloading 10 replays with the exact same size. Each replay size was 10962469 bytes which is approximately 10MB. The 10 replays all together made a total of 109624690 bytes which was approximately 105MB. We measured the time it took to download the 10 replays. We performed two types of download, parallel and sequential. When we downloaded the replays sequentially we downloaded the ten replays one by one, only after the current video finished downloading the next started. In the parallel scenario, the files started downloading at the same time, all at once. We did the whole measurement twice, once with TDLS off and the other with TDLS on. For all the different network configurations we

measured 8 times the time it took to transfer all the replays from one device to the other. The measurements occurred all in the same place under the same conditions. All the devices used in the measurement were in the same room. With distance of at most 2 meters from each other. The remote server was a laptop (ASUS GL552VX running Linux Mint 18.1) which was working as an access point for the edge-cloud too. The services and all the replays were stored in the laptop. The laptop had no other networking applications running by the time the measurements were done. The devices used to form the edge-cloud(s) were all the same (Nexus 9 running Android 6.0.1) and all of them were running the same version of the Android application. They were running no other user applications than UGR at the time the measurement was done.

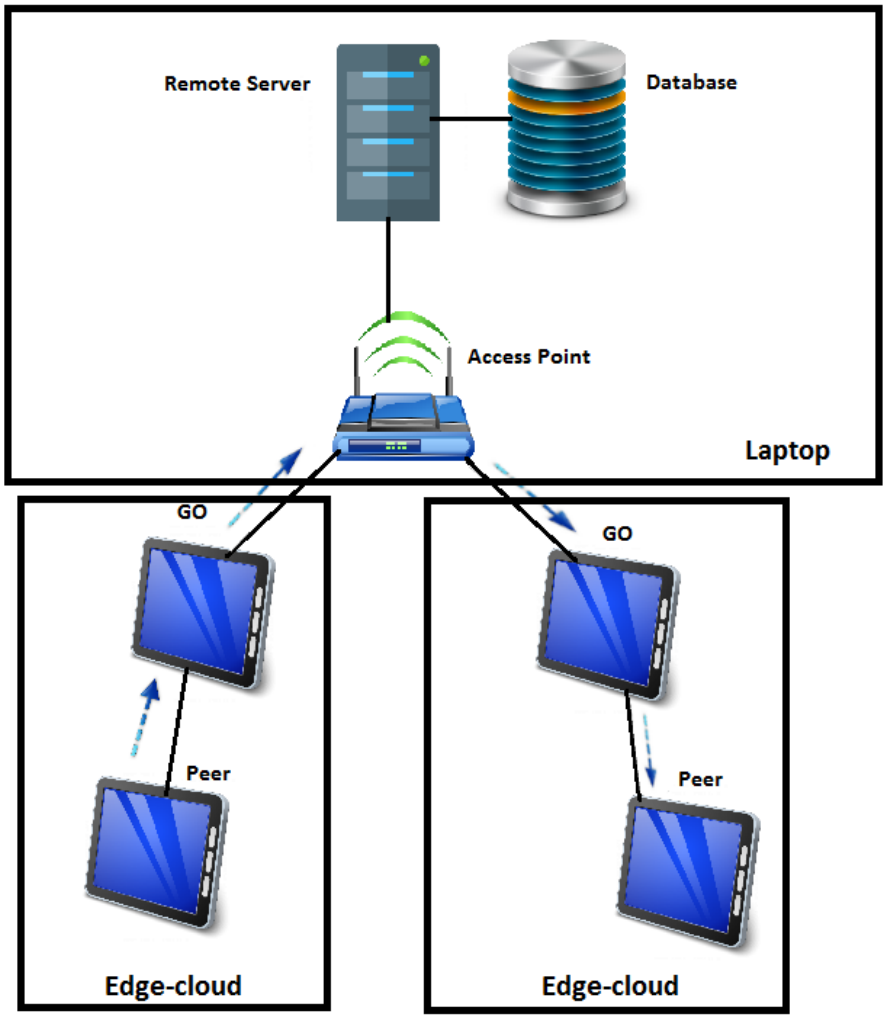


Figure 5.1: Inter-group Peer to Peer

Figure 5.1 shows the network configuration to test the inter-group replay transfer.The replay goes from one Peer to another Peer in a different edge-cloud. The path the content follows starts in a Peer, goes through its Group Owner to the Access Point and the Access Point redirects it to the Group Owner of the receiver edge-cloud which will forward it to the Peer who requested the contents. The Remote Server does not get involved on this procedure.

Figure 5.2 represents a configuration very similar to the previous one except the fact that the replay contents being transmitted among edge-clouds do not come from a Peer inside an edge-cloud but instead from the edge-cloud's Group Owner.

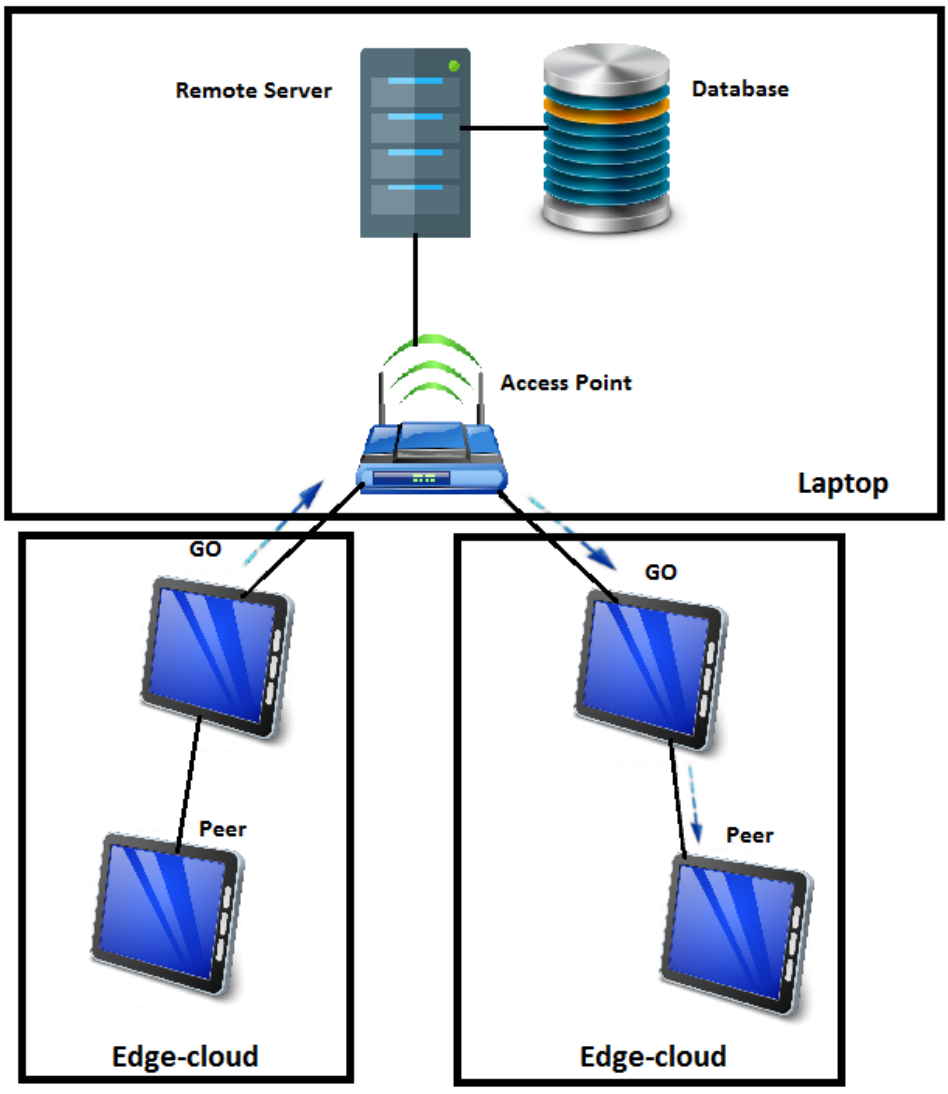


Figure 5.2: Inter-group Group Owner to Peer

Figure 5.3 shows a topology with only one edge-cloud containing a Group Owner with two Peers connected to it. The measurements performed with this topology were motivated by the need for a comparison of the time it takes to transfer a replay between two Peers inside the same group and two Peers in different edge-clouds. The replay goes from a Peer through its Group Owner to the other Peer in the same edge-cloud. Similar to what happens in the previous schemes the server has no active role in this kind of replay transfer.

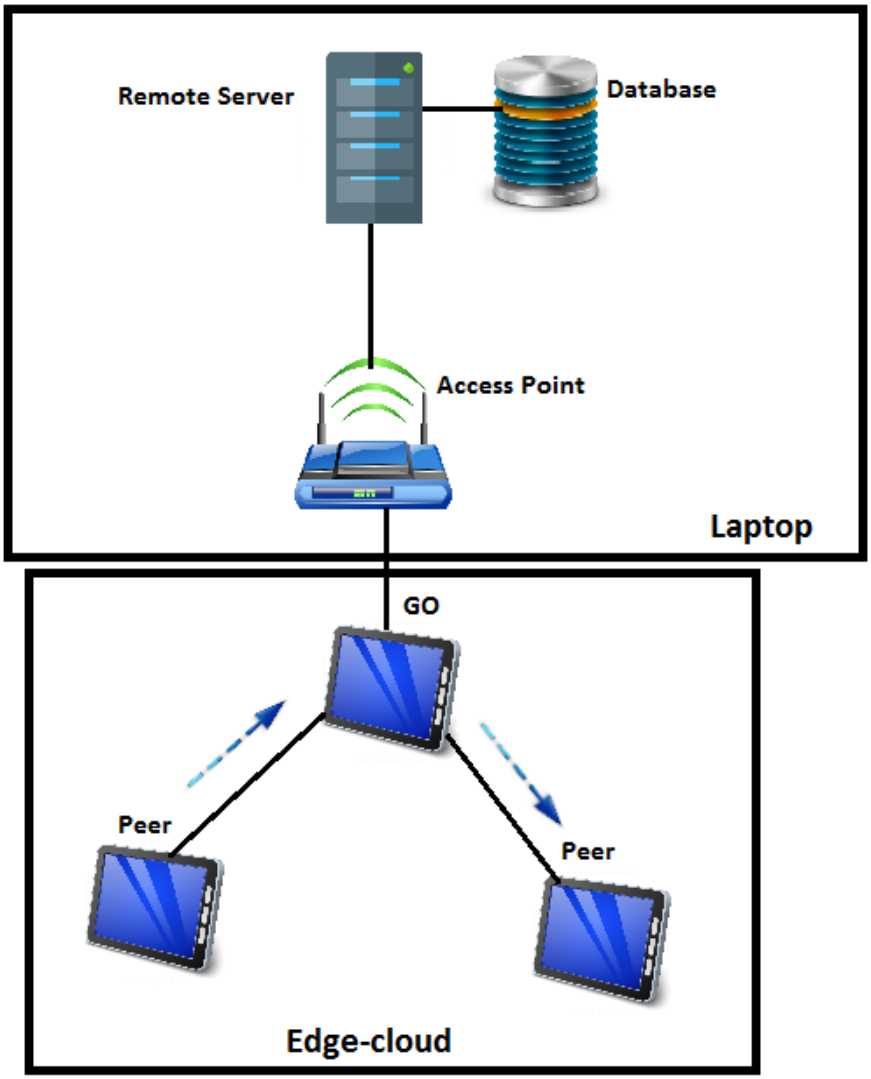


Figure 5.3: Same group, Peer to Peer

Figure 5.4 outlines the same network configuration as the previous one but this time we'll share replays directly between the Group Owner and a Peer in the same group. The motivation for this measurement comes from the need to compare the time lost with one hop. This time the replay goes directly from the Group Owner to the Peer.

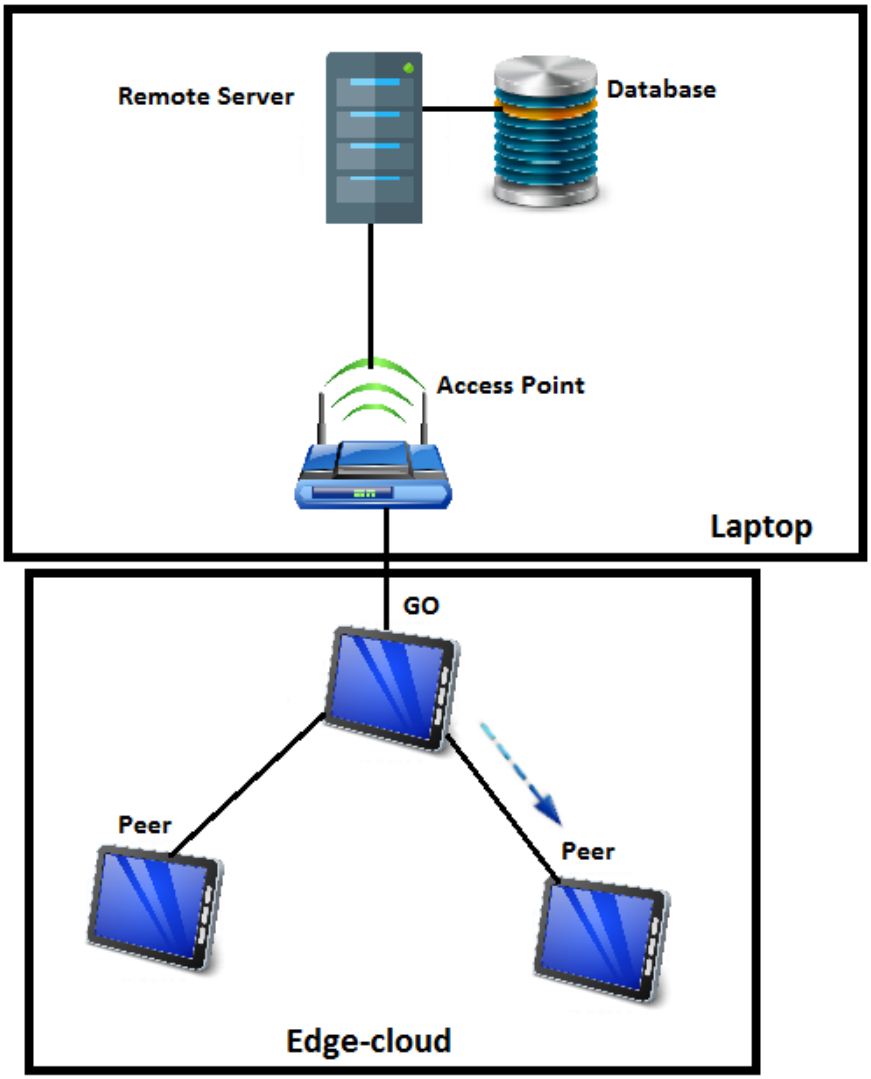


Figure 5.4: Same group, Group Owner to Peer

This time as we can see in Figure 5.5 we just measured the time the replays took to be downloaded from the Remote Server to the closest device in terms of hops, which is the Group Owner of an edge-cloud. By the time the measurement was done there was no other devices connected neither to the Access Point nor to the Group Owner.

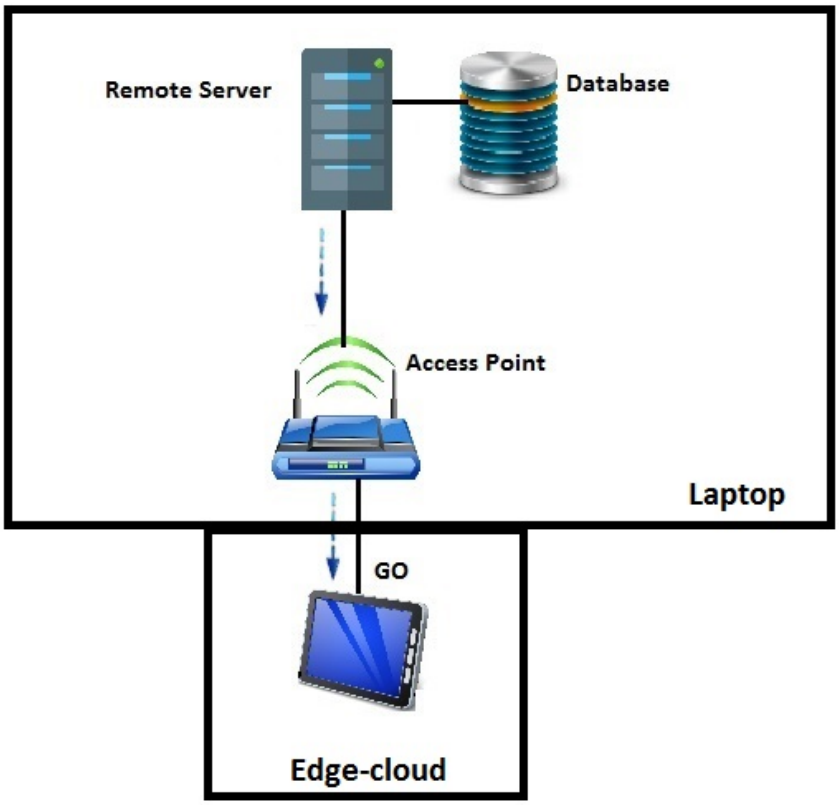


Figure 5.5: Server to Group Owner

Last in Figure 5.6 we can see the outline of a network composed only by a Group Owner connected to the Remote Server and a Peer. The purpose of this measurement was to see how long it will take for a Peer to download a replay from the Remote Server passing by the Group Owner. Similar to the previous scenarios there was no other devices in the network besides the ones outlined in the figure.

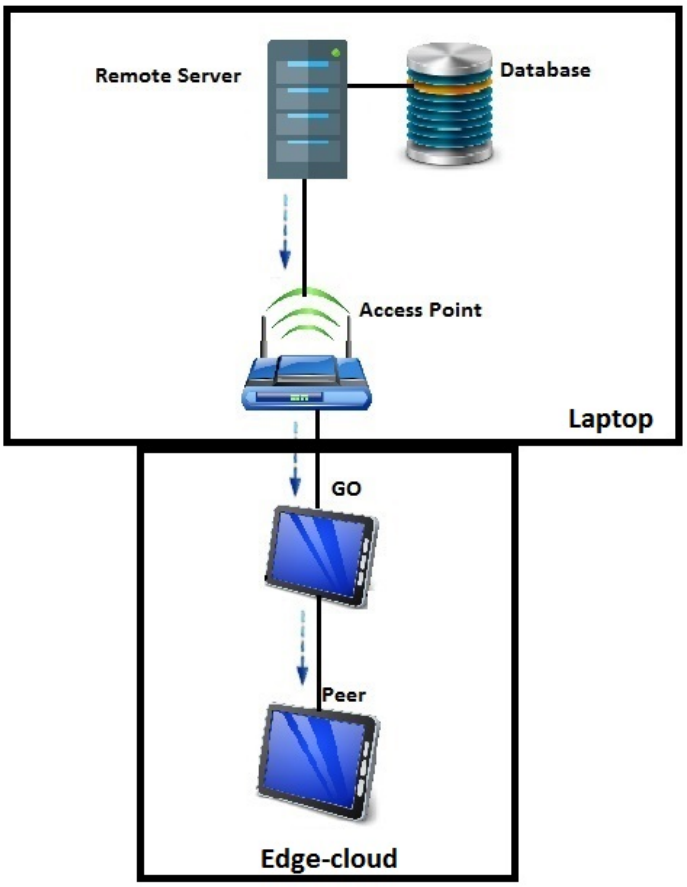


Figure 5.6: Server to Peer

			With TDLS	Margin of Error (95%)	No TDLS	Margin of Error (95%)
Peer to Peer	**Inter Group**	**Parallel**	2.39 MB/s	0.035	2.25 MB/s	0.068
		Sequential	2.46 MB/s	0.035	2.43 MB/s	0.049
	Same Group	**Parallel**	4.70 MB/s	0.221	4.27 MB/s	0.092
		Sequential	6.39 MB/s	0.084	4.66 MB/s	0.082
GO to Peer	**Inter Group**	**Parallel**	3.30 MB/s	0.033	3.40 MB/s	0.025
		Sequential	3.11 MB/s	0.054	2.49 MB/s	0.051
	Same Group	**Parallel**	7.65 MB/s	0.298	7.14 MB/s	0.246
		Sequential	8.65 MB/s	0.242	8.16 MB/s	0.293

Table 5.1: Transfer rates from device to device

		Parallel	Margin of Error (95%)	Sequential	Margin of Error (95%)
From Server to:	**GO**	7.65 MB/s	0.542	9.46 MB/s	0.341
	Peer	3.74 MB/s	0.140	4.10 MB/s	0.116

Table 5.2: Transfer rates from server and devices

5.2.1 Analysis

In the measurements we did not test with more than one device simultaneously downloading contents from other devices at the same time, we just measured the performance attained using only one device per edge-cloud besides the Group Owner except when we measured the transfer rate between devices inside the same edge-cloud. The measurements were made with a confidence interval of 95%. In our measurements the Access Point was only being used by the UGR application, but in real case scenarios the transfer rate among edge-clouds is strongly dependent on the load placed on the Access Point. In terms of load its indifferent for the access point to forward the replays either from the Remote Server or other edge-clouds to the devices that request it, this way its preferable to go fetch it in another edge-cloud rather than to the Remote Server. We may remove a significant amount of load from the Remote Server which will be distributed by the edge-clouds' Group Owners. In every measurement we did we never got a speed below 2.25MB/s. In almost every case the TDLS proved to be helpful increasing overall traffic transfer rate except in the parallel inter-group - Group Owner to Peer scenario. The use of TDLS increased the overall transfer rate among devices in approximately 9%. One of the main reasons why the use of TDLS was shown to be advantageous was because the devices were close to each other, they all were in the same room which favoured the use of TDLS but that might not happen in a real world scenario. When comparing the transfer rate inside the same edge-cloud and between different edge-clouds, the transfer rate between different edge-clouds was approximately 56% slower than inside the same edge-cloud. The sequential method for replay transfer between devices showed to be approximately 6% faster than the parallel one, but when downloading the contents from the server to the devices the transfer rate gap between the two methods was even larger. The file transfer rate from the server to devices using the sequential method was 17% faster than the parallel. The Server showed slightly more trouble dealing with multiple download requests at the same time than the Android devices. Regarding the number of hops a file does we measured the transfer rate between Peer to Peer in the same group (this involves

the traffic to pass by the Group Owner), which makes a total of 2 hops, and from the Group Owner directly to a Peer in the same group which is only 1 hop, 1 less than the previous. The extra hop reduced the transfer rate among devices by approximately 37%. The same experience was made but this time the content transfer was between two different edge-clouds. Transferring content from a Peer from a certain edge-cloud to another Peer in another edge-cloud proved to be approximately 21% slower than transferring the same exact content from the Group Owner of an edge-cloud to a Peer of another edge-cloud which is one hop less. Lastly we measured the difference that 1 hop made when transferring the contents from the Remote Server. Transferring the contents from the Remote Server to a Peer of an edge-cloud proved to be approximately 54% slower than transferring the same exact content to the Group Owner of that same edge-cloud which is 1 hop less. This shows that the Group Owner has some trouble dealing with the reception and forwarding of the requests that come from the Remote Server to another device, which is something that might be worked on in future work.

Chapter 6

Conclusions

During this project we built all the things that we proposed ourselves to do in the beginning. We built an Android application that makes use of the Hyrax middleware and allows users to download and share video replays. We created a database capable of dealing with multiple Events at the same time and allow differed events to be subscribed in the Android application. An administrator interface was built as well for the UGR to allow the administrators to supervise replays, modify and set the event's settings and add new replays to an open event. Last, the possibility for different groups to share contents among themselves was also implemented which allows devices from different edge-clouds connected to the same access point to share content without the need of Internet access. Based on the results gathered in our experiments with the network setups we set we can assume that the content sharing speeds among devices either in the same edge-cloud or between different edge-cloud is enough to make this system viable. We could easily see that the Group Owner is being a bottleneck when transferring the replays directly from the server to a peer connected to the Group Owner. The Group Owner's role here is to simply forward the contents it receives from the Remote Server to the peer. It plays the same exact role when two peers inside the same edge-cloud transfer contents between them, the Group Owner only forwards the contents from one peer to the other but when it forwards the contents it receives from the Remote Server to another peer a significant transfer rate loss occurs. The download speed a peer attains when downloading a replay from the Remote Server is very similar to the speed it attains when it downloads the same exact content from a different edge-cloud. In the communication between peers it would be worthwhile to do caching of the content in the GO both between groups and within the same group. The server experiences a loss of performance when dealing with multiple downloads

simultaneously, possible cause can be the lack of parallelism of Node.js. Something that could now be explored is the possibility to redo the exactly same measurements but this time with a larger number of devices involved in order to create a larger and more congested network so as to make it look more like a real world scenario.

Bibliography

[1] J. a. Rodrigues, E. R. B. Marques, L. M. B. Lopes, and F. Silva, "Towards a middleware for mobile edge-cloud applications," in *Proceedings of the 2Nd Workshop on Middleware for Edge Clouds & Cloudlets*, ser. MECC '17. New York, NY, USA: ACM, 2017, pp. 1:1–1:6.

[2] "global mobile statistics 2016, q2 report," http://mobiforge.com/, online; accessed 04 January 2017.

[3] "Cisco Visual Networking Index: Global Mobile Data Traffic," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html, online; accessed 04 January 2017.

[4] H.-D. Cho, P. D. P. Engineer, K. Chung, and T. Kim, "Benefits of the big. little architecture," *EETimes, Feb*, 2012.

[5] "VLC, a free and open source cross-platform multimedia player," https://play.google.com/store/apps/details?id=org.videolan.vlc, online; accessed 0 January 2017.

[6] "GPS run tracker that combines traditional fitness with mobile applications and social networking," https://www.runtastic.com/, online; accessed 05 January 2017.

[7] "Glucose Buddy - Diabetes Logbook Manager w/syncing, Blood Pressure, Weight Tracking," http://www.glucosebuddy.com/glucose_buddy_app, online; accessed 05 January 2017.

[8] "iCare Blood Pressure Monitor," http://www.icarefit.com/product_pc.htm, online; accessed 05 January 2017.

[9] U. Drolia, R. Martins, J. Tan, A. Chheda, M. Sanghavi, R. Gandhi, and P. Narasimhan, "The case for mobile edge-clouds," *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, 2013.

[10] "Alljoyn Framework," https://allseenalliance.org/framework, online; accessed 05 January 2017.

[11] "Apple's Implementation of Bonjour,," https://www.apple.com/support/bonjour/, online; accessed 05 January 2017.

[12] "OpenGarden's FireChat App," http://opengarden.com/firechat/, online; accessed 05 January 2017.

[13] "Hyrax: Crowd-Sourcing Mobile Devices to Develop Edge Clouds," hyrax.dcc.fc.up.pt/, online; accessed 06 January 2017.

[14] J. Erman and K. Ramakrishnan, "Understanding the super-sized traffic of the super bowl," *Proceedings of the 2013 conference on Internet measurement conference - IMC '13*, 2013.

[15] J. M. Rodríguez, C. Mateos, and A. Zunino, "Are smartphones really useful for scientific computing?" *Advances in New Technologies, Interactive Interfaces and Communicability Lecture Notes in Computer Science*, p. 38–47, 2012.

[16] "MPI: A message passing interface," in *Supercomputing '93. Proceedings*, Nov 1993, pp. 878–883.

[17] D. C. Doolan, S. Tabirca, and L. T. Yang, "MMPI a message passing interface for the mobile environment," *Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia - MoMM '08*, 2008.

[18] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto clouds: Leveraging mobile devices to provide cloud service at the edge," *2015 IEEE 8th International Conference on Cloud Computing*, 2015.

[19] E. Miluzzo, R. Cáceres, and Y.-F. Chen, "Vision: mClouds - computing on clouds of mobile devices," *Proceedings of the third ACM workshop on Mobile cloud computing and services - MCS '12*, 2012.

[20] N. Fernando, S. W. Loke, and W. Rahayu, "Honeybee: A programming framework for mobile crowd computing," *Lecture Notes of the Institute for Computer*

Sciences, Social Informatics and Telecommunications Engineering Mobile and Ubiquitous Systems: Computing, Networking, and Services, p. 224–236, 2013.

[21] "Multipeerconnectivity - support peer-to-peer connectivity and the discovery of nearby devices." https://developer.apple.com/reference/multipeerconnectivity, online; accessed 06 January 2017.

[22] S. S. Kanhere, "Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces," *2011 IEEE 12th International Conference on Mobile Data Management*, 2011.

[23] A. Tamilin, I. Carreras, E. Ssebaggala, A. Opira, and N. Conci, "Context-aware mobile crowdsourcing," *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12*, 2012.

[24] "Protocol buffers, a language-neutral, platform-neutral extensible mechanism for serializing structured data," https://developers.google.com/protocol-buffers/, online; accessed 05 January 2017.